

AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions and listings of claims in the application.

1. (currently amended) A method for providing content, comprising the steps of: receiving a request from a user device for particular content, said the request is received at a server;

accessing a mark-up markup language description of said the particular content, said the mark-up markup language description includes markup language one or more source code files which describes describes a behavior of said the particular content on a user interface of said the user device based on user interactions with the particular content via the user interface, said the mark-up markup language description includes a reference to a view instance element, the view instance element includes a reference to data for rendering on said user interface, said mark-up language description defines a connection to an external data source for said data, said external data source is external to said server;

accessing said data at said external data source based on said mark-up language description, said server performs said accessing;

at the server, converting the markup language description to a script source code by:
compiling said mark-up language description of said particular content to create executable code for said user device, said step of compiling is performed at said server in response to said request, the compiling the mark-up language description includes:

accessing the markup language source code to locate a tag name and associated attributes for the view instance element, the source code includes a tag name and attributes; and

creating script source code based on the markup language source code, the script source code includes a script instruction to call an instantiation view function in a function call, the script instruction includes adding the tag name and associated attributes in to the function call as parameters, the instantiation view function determines, based on the parameters, what kind of objects the user device creates at a runtime of the executable code at the user device; and

providing executable code for the user device by compiling the script source code function call-to byte code;, the step of compiling is performed at the server in response to the request; and transmitting said the executable code and said data from said the server to said the user device for execution by said the user device to allow a user to access said the particular content and said data via said the user interface according to said the behavior and said the user interactions, the instantiation view function is called at a runtime of the executable code at the user device, and creates objects which are displayed on the user interface, based on the tag name and associated attributes.

2. (cancelled)

3. (cancelled)

4. (currently amended) A method according to claim 1, wherein:

said the user device includes a rendering entity and a browser, said the rendering entity is a plug-in to said the browser, said the plug-in is embedded in said the browser before said the request, and said the rendering entity executes said the executable code.

5. (currently amended) A method according to claim 1, wherein:

said step of compiling includes parsing the mark-up language description to identify elements of the mark-up language description, translating the elements to the script source code comprises ActionScript and the byte code comprises corresponding and compiling said ActionScript into ActionScript byte code.

6. (cancelled)

7. (currently amended) A method according to claim 1, wherein:

said the request for particular content is received from a browser in which a rendering entity is present as a plug-in to said the browser, said the browser is at said the user device, and said the rendering entity executes said the executable code.

8. (currently amended) A method according to claim 1, wherein:
~~the data-particular content comprises a media file content comprising video; the markup language source code includes a source attribute which references a name of the media file, the source attribute is within a window tag in the markup language source code; and— said the media file content is transmitted with said the executable code from said the server to said the user device for use by a rendering entity at said the user device in allowing the user to access said the video in a window on said the user interface when said the media content file is referenced by the source attribute within the window tag when said the executable code is executed.~~

9. (cancelled)

10. (cancelled)

11. (currently amended) A method according to claim 1, further comprising the step of: authenticating ~~said the request, said the steps of compiling and transmitting are only performed if said the step of authenticating is successful, and different types of authenticating are provided for at least one of: a) different types of content, including content types of application, data and service, and b) each item of content.~~

12. (cancelled)

13. (currently amended) A method according to claim 1, wherein the request from the user device for ~~said the~~ particular content is a request for a first application, the first application runs on ~~said the~~ user device when ~~said the~~ executable code is executed at ~~said the~~ user device, the method further comprising the steps of:

receiving a request at ~~said the~~ server from ~~said the~~ first application running on the user device for a second application;

in response to the request, accessing and compiling the second application; and

transmitting ~~said the~~ compiled second application from ~~said the~~ server to ~~said the~~ user device.

14.-27. (cancelled)

28. (currently amended) One or more processor readable storage devices having processor readable code embodied on said the processor readable storage devices, said the processor readable code for programming one or more processors to perform a method comprising the steps of:

receiving a request for particular content from a browser, said the browser having a plug-in embedded therein, said the request is received at a server;

in response to the request, accessing a mark-up markup language description of said the particular content, said the mark-up markup language description uses a source attribute to reference references a name of a media file comprising at least one of audio, video and a movie;

compiling said the mark-up markup language description of said the particular content to create executable code for said the plug-in to said the browser, said the executable code provides said the particular content, and said the step of compiling is performed at said the server in response to said the request; and

transmitting said the executable code and said the media file from said the server to said the plug-in, said the plug-in renders said the particular content on a user interface based on said the executable code, the source attribute and said the media file.

29.-32. (cancelled)

33. (currently amended) One or more processor readable storage devices having processor readable code embodied on said the processor readable storage devices, said the processor readable code for programming one or more processors to perform a method comprising the steps of:

receiving a request for particular content, said the request is received at a server from a web client in which a plug-in is embedded;

accessing a mark-up markup language description of said the particular content, said the mark-up markup language description includes markup language one or more source code files which describe describes a behavior of said the particular content on a user interface of said the web client based on dynamic user interactions with the particular content via the user interface, the

markup language description includes a reference to a view instance element;

in response to said the request, converting the markup language description to a script source code by:

compiling said mark-up language description to create executable code for said plug-in to said web client, said executable code implements said user interface, said step of compiling is performed at said server in response to said request, the compiling the mark-up language description includes:

accessing the markup language source code to locate a tag name and associated attributes for the view instance element, the source code includes a tag name and attributes; and

creating creating a script source code based on the markup language source code, the script source code includes a script instruction to call an instantiation view function in a function call, the script instruction includes the tag name and associated attributes in the function call as parameters;

add the tag name and attributes to the function call as parameters, the instantiation view function determines, based on the parameters, what kind of objects the plug-in creates at a runtime of the executable code at the user device; and

compiling the function call to byte code; and

compiling the script source code to byte code, the step of compiling is performed at the server in response to the request;

providing executable code for the user device which includes the byte code; and

transmitting said the executable code from said the server to said the plug-in for execution by said the plug-in to allow a user to access and dynamically interact with said the particular content via said the user interface, the instantiation view function is called at a runtime of the executable code at the plug-in, and creates objects which are displayed on a user interface, based on the tag name and associated attributes.

34.-40. (cancelled)

41. (currently amended) An apparatus, comprising:

one or more storage devices; and

one or more processors in communication with said the one or more storage devices, said the one or more processors: (a) receive a request for particular content from an HTTP client having a plug-in embedded therein, said the request is received at a server, (b) access a mark-up markup language description of said the particular content, said the mark-up markup language description describes a behavior of said the particular content on a user interface of said the HTTP client based on user interactions with the particular content via the user interface, said the mark-up markup language description includes a reference to a view instance element, the view instance element includes a reference to data for rendering on said the user interface; and (c) compile said the mark-up markup language description of said the particular content to create executable code for said the plug-in to said the HTTP client, wherein the one or more processors, to compile the compiling the mark-up markup language description-includes:

access accessing source code for the view instance element, the source code includes a tag name and attributes;

create a script instruction to call an instantiation view function in a function call;

add the tag name and attributes to the function call as parameters, the instantiation view function determines, based on the parameters, what kind of objects the user device creates at a runtime of the executable code at the user device; and

compile compiling the function call to byte code; and

said the executable code provides said the particular content, said the compiling is performed at said the server in response to said the request, and said the executable code is transmitted from said the server to said the plug-in for execution by said the plug-in to provide said the particular content via said the user interface according to said the behavior and said the user interactions.

42. (cancelled)

43. (cancelled)

44. (currently amended) An apparatus according to claim 41, wherein the view instance element includes a source attribute which provides a reference to a file name of media content, and

the one or more processors;

access the media content using the source attribute;

create an object representation of the media content, the object representation includes the media content and fields which store attributes of the media content, the attributes include a name of the media content and a format of the media content;

remove the media content from the object representation and insert a reference to the media content into the object representation in place of the media content, then compile the object representation to byte code;

create a tag header;

add the media content, but not the compiled object representation, to the tag header; and

transmit the compiled object representation with the compiled mark-up markup language description from said the server to said the user device for execution by said the user device to provide said the particular content and said the data via said the user interface according to said the behavior and said the user interactions.

45. (currently amended) An apparatus, comprising:

one or more storage devices; and

one or more processors in communication with said the one or more storage devices, said the one or more processors; perform a method comprising the steps of:

receive reeceiving a request for particular content, said the request is received at a server, said the request is from a client which includes a browser and a rendering engine that is different than said the browser but operates in connection with said the browser;

access accessing-first code associated with said the particular content at said the server, said the first code comprises elements that are identified by markup language tags, at least one of said the elements references a media content external to said the server;

acccss eeeesing the media content;

at the server, in response to the request, compile compiling said the first code to create executable code for said the rendering engine, said step of compiling is performed at said server in response to said request;

create creating an object representation of the media content, the object

representation includes the media content and fields which store attributes of the media content, the attributes include a name of the media content and a format of the media content;

remove removing the media content from the object representation and inserting a reference to the media content into the object representation in place of the media content, then compiling the object representation to byte code;

create creating a tag header;

add adding the media content, but not the compiled object representation, to the tag header; and

transmit ~~transmitting~~ said the executable code with the compiled object representation from said the server to said the client for rendering of said the particular content and the media content by said the rendering engine, said the executable code implements a user interface at said the client that provides access to said the particular content.

46. (cancelled)

47. (currently amended) An apparatus according to claim 45, wherein the one or more processors implement a media transcoder which includes interfaces for images, audio, graphics and video, and said the media transcoder transcodes said the media content to an accepted format before the media content is added to the tag header, said the transcoding is separate from said the compiling of the first code. [.]

48.-51. (cancelled)

52. (currently amended) A method according to claim 4, wherein:

the server has separate object code generators and compilers for different types of rendering entities;

said the request is received at said the server from said the user device and includes an indication that identifies a type of said the rendering entity from among the different types of

rendering entities; and

said compiling includes creating said the executable code is created specific for said the type of rendering entity using corresponding ones of the object code generators and compilers, in response to said the indication.

53. (currently amended) A method according to claim 1, wherein:
said the executable code comprises one or more binary files.

54. (currently amended) A method according to claim 1, wherein:
said the executable code comprises at least one of object code and byte code.

55. (cancelled)

56. (currently amended) One or more processor readable storage devices according to claim 33, 55, wherein:

the markup language description comprises elements which are identified by markup language tags; and

at least one of said the elements defines a view template of a user interface element, said the view template is instantiated when said the executable code is executed by said the rendering entity.

57. (currently amended) One or more processor readable storage devices according to claim 56, wherein:

said the elements comprise at least one element which defines a view class which supplies default properties, behavior, and child views which the view template instantiates, the child views are associated with a parent view.

58. (currently amended) One or more processor readable storage devices according to claim 33, 55, wherein:

the markup language description comprises elements which are identified by markup language tags; and

at least one of said the elements provides a source tag which references a media file comprising at least one of an animation, audio, video and a movie.

59.-61. (cancelled)

62. (currently amended) One or more processor readable storage devices according to claim 28, wherein:

said the media file comprises a .SWF file, said the markup language description includes a tag that references a file name of said the .SWF file, within a window tag, so that the .SWF file is played within a window on the user interface when the plug-in renders the particular content.

63. (cancelled)

64. (currently amended) One or more processor readable storage devices according to claim 33, 55, wherein:

the markup language description comprises elements which are identified by markup language tags; and

at least one of said the elements provides an inline definition of formatted text.

65. (currently amended) One or more processor readable storage devices according to claim 33, 55, wherein:

the markup language description comprises elements which are identified by markup language tags; and

at least one of said the elements provides an inline definition of vector graphics using an svg tag and associated attributes within a window tag, in response to which the vector graphics are rendered in a window on the user interface when the executable code is executed.

66.-68. (cancelled)

69. (currently amended) A method according to claim 1, wherein:

said the compiling comprises parsing said the markup language description to identify first and second types of elements in the markup language description, providing said the first type of element to a first compiling module which is appropriate for said the first type of element to obtain first object code, providing said the second type of element to a second compiling module which is appropriate for said the second type of element to obtain second object code, and assembling said the first and second object code into a single executable; and

said the transmitting said the executable code comprises transmitting said the single executable to said the user device.

70. (currently amended) A method according to claim 69, wherein:

said the first type of element provides a script which defines said the behavior of said the particular content, and said the second type of element defines a said-connection to an said external data source.

71. (cancelled)

72. (cancelled)

73. (currently amended) A method according to claim 4, wherein:
said the rendering entity is a Flash player.

74. (cancelled)

75. (cancelled)

76. (cancelled)

77. (currently amended) One or more processor readable storage devices according to claim 33, 55, wherein:

the markup language description comprises elements which are identified by markup

language tags; and

said the elements comprises elements which define script code, said the script code specifies a visual appearance of said the user interface when the executable code is executed.

78. (currently amended) One or more processor readable storage devices according to claim 33, 55; wherein:

the markup language description comprises elements which are identified by markup language tags; and

said the elements comprises elements which define script code, said the script code specifies an application logic of said the mark-up markup language description.

79. (currently amended) One or more processor readable storage devices according to claim 33, 55; wherein:

the markup language description comprises elements which are identified by markup language tags; and

said the elements comprises elements which define script code, said the script code specifies a connection to an external data source, said the external data source includes data for rendering on said the user interface by said the plug-in.

80. (currently amended) A method according to claim 28, wherein:

said the plug-in is a Flash player.

81. (currently amended) A method according to claim 8, further comprising:
providing an object in the executable code which includes fields storing attributes which identify identifies a the name and a format of the media file content, the name and format are provided via the user interface when said the media file content is rendered.

82. (currently amended) A method according to claim 8, wherein:

said the request for particular content is received from a browser in which a plug-in to said the browser is present, said the browser is at said the user device, and said the plug-in renders said

the media file content.

83. (currently amended) A method according to claim 1, wherein:

said the executable code provides a script based on the script source code which is executed when a specified event occurs when a user interacts with the particular content via the user interface, the specified event is based on at least one of user control of a pointing device and or a key press.

84. (cancelled)

85. (currently amended) A method according to claim 1, wherein:

the instantiation view function is predefined, and the executable code, when executed at the user device, calls the a predefined instantiation view function associated with the tag name, and passes the attributes to the predefined instantiation view function.

86. (currently amended) A method according to claim 1, wherein:

the instantiation view function is predefined, and the executable code, when executed at the user device, calls a the user-defined instantiation view function associated with the tag name, and passes the attributes to the user-defined instantiation view function.

87. (currently amended) A method according to claim 1, wherein the particular content includes data comprises media content, the method further comprising:

receiving the media content at the server, from an the external data source;

creating an object representation of the media content, the object representation includes the media content and fields which store attributes of the media content, the attributes include a name of the media content and a format of the media content;

removing the media content from the object representation and inserting a reference to the media content into the object representation in place of the media content, then compiling the object representation to byte code;

creating a tag header;

adding the media content, but not the compiled object representation, to the tag header; and

transmitting the compiled object representation with the compiled mark-up markup language description from said the server to said the user device for execution by said the user device to provide said the particular content and said the data via said the user interface according to said the behavior and said the user interactions.

88. (currently amended) A method according to claim 87, further comprising:
assembling the compiled mark-up markup language description and the compiled object representation into a single executable which is transmitted as said the executable code from said the server to said the user device.

89. (currently amended) A method according to claim 87, further comprising:
transcoding said the media content before the adding the media content to the tag header, said the transcoding is separate from the compiling of the object, the media content comprises audio which is transcoded between MP3 and WAV.

90. (currently amended) A method according to claim 87, further comprising:
transcoding said the media content before the adding the media content to the tag header, said the transcoding is separate from the compiling of the object, the media content comprises video which is transcoded between any two of the following formats: MPEG, MPEG2, SORENSEN, REAL, and Animated GIF.

91. (currently amended) A method according to claim 87, further comprising:
transcoding said the media content before the adding the media content to the tag header, said the transcoding is separate from the compiling of the object, the media content comprises an image which is transcoded between any two of the following formats: JPEG, GIF, BMP, and PNG.

92. (currently amended) A method according to claim 87, further comprising:
transcoding said the media content before the adding the media content to the tag header, said the transcoding is separate from the compiling of the object, the media content comprises graphics which is transcoded between any two of the following formats: SVG, HTML, PDF and

SWF.

93. (previously presented) A method according to claim 87, wherein: the media content comprises audio.

94. (previously presented) A method according to claim 87, wherein: the media content comprises video.

95. (previously presented) A method according to claim 87, wherein: the media content comprises a movie.

96. (previously presented) A method according to claim 87, wherein: the media content comprises an animation.

97. (currently amended) A method according to claim 87, wherein: said the media content comprises a .SWF file.

98. (currently amended) One or more processor readable storage devices according to claim 33, wherein the view instance element includes a reference to media content, and the method performed further comprises:

accessing the media content;

creating an object representation of the media content, the object representation includes the media content and fields which store attributes of the media content, the attributes include a name of the media content and a format of the media content;

removing the media content from the object representation and inserting a reference to the media content into the object representation in place of the media content, then compiling the object representation to byte code;

creating a tag header;

adding the media content, but not the compiled object representation, to the tag header; and transmitting the compiled object representation with the compiled mark-up markup language

description from said the server to said the plug-in for execution by said the plug-in to provide said the particular content and said the data via said the user interface according to said the behavior and said the user interactions

99. (currently amended) One or more processor readable storage devices according to claim 98, wherein the method performed further comprises:

assembling the compiled mark-up markup language description and the compiled object representation into a single executable which is transmitted as said the executable code from said the server to said the plug-in.

100. (currently amended) One or more processor readable storage devices according to claim 98, wherein the method performed further comprises:

transcoding said the media content before the adding the media content to the tag header, said the transcoding is separate from the compiling of the object.

101. (previously presented) An apparatus according to claim 45, wherein:
the media file comprises at least one of audio, video and a movie.

102. (currently amended) An apparatus according to claim 45, wherein the compiling of the first code comprises:

accessing source code for the at least one of said the elements which references the media file, the source code includes a tag name and attributes;

create a script instruction to call an instantiation view function in a function call;

add the tag name and attributes to the function call as parameters, the instantiation view function determines, based on the parameters, what kind of objects the client creates at a runtime of the executable code at the client; and

compiling the function call to byte code.

103. (new) A method according to claim 1, wherein:
the objects which are created by the instantiation view function, and displayed on the user

interface, include at least one of windows, dialogs, buttons, images, text fields, banners and animation.

104. (new) A method according to claim 1, wherein:

the objects which are created by the instantiation view function, and displayed on the user interface, include at least one of dialogs, buttons, and text fields.

105. (new) A method according to claim 1, wherein:

the instantiation view function, when called at the runtime of the executable code at the user device, creates a sound object on the user device, based on the tag name and associated attributes.

106. (new) A method according to claim 1, wherein:

the tag name is parent tag name;

the converting the markup language description to the script source code includes accessing the markup language source code to locate an associated child tag name and associated attributes in the view instance element;

the script instruction includes the child tag name and associated attributes in the function call as parameters;

the instantiation view function, when called at the runtime of the executable code at the user device, creates the objects which are displayed on the user interface, based on the child tag name and associated attributes.

107. (new) A method according to claim 106, wherein:

the instantiation view function is called recursively at the runtime of the executable code at the user device, first with the parent tag and associated attributes and then with the child tag and associated attributes, and results from calling the instantiation view function with the child tag and associated attributes are attached to results from calling the instantiation view function with the parent tag and associated attributes.

108. (new) A method according to claim 1, wherein:

when the instantiation view function is called at the runtime, a class which is associated with the tag name is identified, and a separate table of instantiation functions for the class, indexed by tag name, is accessed, to create the objects which are displayed on the user interface.

109. (new) A method according to claim 1, wherein:

when the instantiation view function is called at the runtime, a table of instantiation functions, indexed by tag name, is accessed, to create the objects which are displayed on the user interface.

110. (new) A method according to claim 1, wherein:

the view instance element includes a reference to data for rendering on the user interface, and the markup language description defines a connection to an external data source for the data, the external data source is external to the server; and

the server accesses the data at the external data source based on the markup language description.

111. (new) One or more processor readable storage devices according to claim 28, wherein:

the markup language description uses the source attribute within an image tag, and the media file comprises an image file.

112. (new) One or more processor readable storage devices according to claim 28, wherein:

the markup language description uses the source attribute within a window tag, in response to which the plug-in renders the media file in a window on the user interface.

113. (new) One or more processor readable storage devices according to claim 28, wherein:

the markup language description uses the source attribute within an swf tag, and the media file is a .SWF file.